

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2003-330992

(43)Date of publication of application : 21.11.2003

(51)Int.Cl.

G06F 17/50

(21)Application number : 2003-100239

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>

(22)Date of filing : 03.04.2003

(72)Inventor : MELLORS WILLIAM K  
RICH MARVIN J

(30)Priority

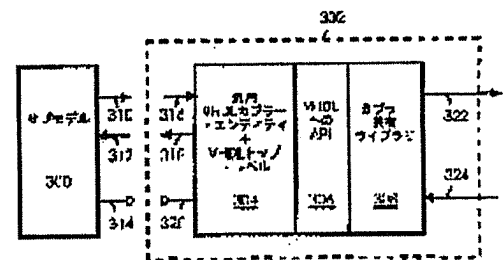
Priority number : 2002 125198    Priority date : 18.04.2002    Priority country : US

## (54) MODEL SIMULATION

(57)Abstract:

**PROBLEM TO BE SOLVED:** To provide techniques for tightly functionally coupling submodels of a partitioned model for concurrent distributed submodel simulation runs.

**SOLUTION:** A technique for distributed processing a partitioned model is provided based on tight functional coupling of multiple submodels of the model. The technique includes, in one embodiment, providing each submodel 300 with a generic coupler 302 to enable processing of the submodel 300 on any simulator instance of any simulator. Submodels 300 coupled with the generic couplers 302 can be processed on the same or different coupling units. The generic couplers 302 facilitate communication between submodels 300 through a common communication directory (CCD) by using functions of a generic coupler shared library 308. The generic couplers 302 further use functions of the shared library 308 to ensure integrity of data transmitted between submodels 300.



(19) 日本国特許庁 (J P)

## (12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2003-330992

(P2003-330992A)

(43) 公開日 平成15年11月21日 (2003. 11. 21)

(51) Int.Cl.<sup>7</sup>

G 0 6 F 17/50

識別記号

6 6 4

F I

G 0 6 F 17/50

テ-マ-ト\* (参考)

6 6 4 L 5 B 0 4 6

審査請求 有 請求項の数48 O L (全 19 頁)

(21) 出願番号 特願2003-100239(P2003-100239)

(22) 出願日 平成15年4月3日 (2003. 4. 3)

(31) 優先権主張番号 10/125198

(32) 優先日 平成14年4月18日 (2002. 4. 18)

(33) 優先権主張国 米国 (US)

(71) 出願人 390009531

インターナショナル・ビジネス・マシー  
ズ・コーポレーションINTERNATIONAL BUSIN  
ESS MACHINES CORPO  
RATION

アメリカ合衆国10504、ニューヨーク州

アーモンク ニュー オーチャード ロー  
ド

(74) 代理人 100086243

弁理士 坂口 博 (外2名)

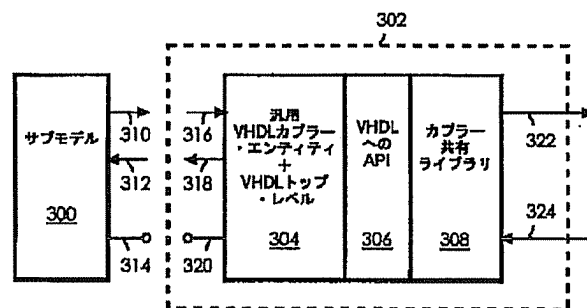
最終頁に続く

(54) 【発明の名称】 モデル・シミュレーション

(57) 【要約】

【課題】 分割されたモデルのサブモデル群をコンカレン  
ト分散サブモデル・シミュレーション実行できるように  
緊密かつ機能的に結合する手法を提供する。

【解決手段】 分割されたモデルの複数のサブモデルを  
緊密かつ機能的に結合することにより分割されたモデル  
を分散処理する手法を提供する。この手法は、一例で  
は、任意のシミュレータの任意のシミュレータ・インス  
タンス上でサブモデル300の処理を可能にする汎用カ  
プラー302を各サブモデル300に備えるステップを  
備えている。汎用カプラー302に結合されたサブモデ  
ル300は同じまたは異なるコンピューティング・ユニ  
ットで処理することができる。汎用カプラー302は汎  
用カプラー共有ライブラリ308の機能を使うことによ  
り、共通通信ディレクトリ (CCD) を通じたサブモデ  
ル300間の通信を容易にする。汎用カプラー302は  
さらに、汎用カプラー共有ライブラリ308の機能を使  
ってサブモデル300間で転送されるデータの一貫性を  
保証する。



## 【特許請求の範囲】

【請求項1】 分割されたモデルの、第1のサブモデルおよび第2のサブモデルを含む複数のサブモデルを処理する方法であって、

分割されたモデルの第1のサブモデルと第2のサブモデルとを結合するステップであって、

前記第1のサブモデルがインタフェースする第1のカプラーを準備するステップと、

前記第2のサブモデルがインタフェースする第2のカプラーを準備するステップと、

前記第1のカプラーと前記第2のカプラーとをインタフェースさせるステップであって、前記第1のカプラーと前記第2のカプラーとの間にバッファを準備するステップとを備えたステップと、

前記第1のサブモデルおよび前記第2のサブモデルを一括して処理するステップであって、前記第1のサブモデルおよび前記第2のサブモデルが前記第1のカプラー、前記バッファ、および前記第2のカプラーを用いて通信する、ステップとを備えたステップを備えた方法。

【請求項2】 前記モデルが論理設計である、請求項1に記載の方法。

【請求項3】 前記一括して処理するステップが前記第1のサブモデルおよび前記第2のサブモデルを分散処理するステップを備え、

前記分散処理するステップが分散プロセス制御を備えている、請求項1に記載の方法。

【請求項4】 前記一括して処理するステップがさらに前記第1のサブモデルを第1のシミュレータ・インスタンス上で処理するステップと前記第2のサブモデルを第2のシミュレータ・インスタンス上で処理するステップとを備え、

前記第1のシミュレータ・インスタンスは前記第2のシミュレータ・インスタンスと異なり、

前記第1のシミュレータ・インスタンスは前記バッファを用いて前記第2のシミュレータ・インスタンスと通信する、請求項1に記載の方法。

【請求項5】 前記第1のシミュレータ・インスタンスは第1のコンピューティング・ユニットに常駐し、前記第2のシミュレータ・インスタンスは第2のコンピューティング・ユニットに常駐する、請求項4に記載の方法。

【請求項6】 少なくとも1つのサブモデルおよび付随するカプラーが少なくとも1つのマップされたデータ・ポートおよびクロック／サイクル・ポートを備え、前記クロック／サイクル・ポートは少なくとも1つのクロック／サイクル信号を供給し、

前記少なくとも1つのクロック／サイクル信号は前記少なくとも1つのサブモデルと付随するカプラーとの間で前記少なくとも1つのマップされたデータ・ポートを通じてデータを転送するのに使用される、請求項1に記載の方法。

【請求項7】 前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方がトップ・レベル・カプラー・エンティティとカプラー共有ライブラリとを備え、前記第1のカプラーおよび前記第2のカプラーのうちの前記少なくとも一方が前記トップ・レベル・カプラー・エンティティを用いて前記第1のサブモデルおよび前記第2のサブモデルと個別に接続し、

前記トップ・レベル・カプラー・エンティティが少なくとも1つの入力と、少なくとも1つの出力と、前記第1のサブモデルおよび前記第2のサブモデルのうちの少なくとも一方の前記クロック／サイクル・ポートにマップされた複数のポートとを備え、

前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方が前記カプラー共有ライブラリを用いて前記カプラーと前記バッファとの間でデータを転送する、請求項6に記載の方法。

【請求項8】 前記バッファが通信媒体を備え、前記通信媒体は前記第1のカプラーと前記第2のカプラーとの間でデータを渡すとともに監視する少なくとも1つのプロトコルを備えている、請求項1に記載の方法。

【請求項9】 前記一括して処理するステップがさらに、前記第1のカプラーが前記第1のサブモデルから取得したデータを前記バッファに送信し、前記第2のカプラーが前記データを前記バッファから取得し、前記データを前記第2のサブモデルに供給するステップと、前記送信と前記取得とを対応させることにより、前記データの一貫性を保証するステップとを備えた、請求項1に記載の方法。

【請求項10】 前記一貫性を保証するステップが、前記データにチェックサム値を挿入するステップと、前記第1のカプラーおよび前記第2のカプラーのうちの一方が前記バッファ中の前記データを調べて前記データの正確性を判断するステップと、前記データが不正確な場合、前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方が、前記データの前記バッファへの再送信および前記データの前記バッファからの再読み取りのうちの少なくとも一方を行なうステップとを備えた、請求項9に記載の方法。

【請求項11】 前記挿入するステップが、前記チェックサム値を計算するステップを備え、前記計算するステップが、少なくとも1つの論理操作を前記データに適用するステップを備えている、請求項10に記載の方法。

【請求項12】 前記調べるステップが、前記データの新たなチェックサム値を計算するステップと、前記新たなチェックサム値と前記チェックサム値とを比較するステップとを備えた、請求項10に記載の方法。

【請求項13】 前記一貫性を保証するステップがさらに、

前記バッファに、前記第1のカプラーが書き込みアクセス可能であるとともに前記第2のカプラーが読み取りアクセス可能な第1の一連ファイルを準備するステップと、

前記バッファに、前記第1のカプラーが読み取りアクセス可能であるとともに前記第2のカプラーが書き込みアクセス可能な第2の一連ファイルを準備するステップと、

所定の順序で、前記第1のカプラーが前記第1の一連ファイルに書き込み、前記第2のカプラーが前記第1の一連ファイルから読み取るステップと、

前記所定の順序で、前記第2のカプラーが前記第2の一連ファイルに書き込み、前記第1のカプラーが前記第2の一連ファイルから読み取るステップとを備えた、請求項9に記載の方法。

【請求項14】前記インタフェースさせるステップがさらに、前記バッファ中のデータを監視して監視統計を供給するステップを備え、

前記監視統計が前記第1のサブモデルおよび前記第2のサブモデルを統計的に再分割して前記一括処理を最適化するのを容易にする、請求項1に記載の方法。

【請求項15】前記モデルがVHDLで表されている、請求項1に記載の方法。

【請求項16】分割されたモデルの、第1のサブモデルおよび第2のサブモデルを含む複数のサブモデルを処理するシステムであって、

分割されたモデルの第1のサブモデルと第2のサブモデルとを結合する手段であって、

前記第1のサブモデルがインタフェースする第1のカプラーを準備する手段と、

前記第2のサブモデルがインタフェースする第2のカプラーを準備する手段と、

前記第1のカプラーと前記第2のカプラーとをインタフェースさせる手段であって、前記第1のカプラーと前記第2のカプラーとの間にバッファを準備する手段とを備えた手段と、

前記第1のサブモデルおよび前記第2のサブモデルを一括して処理する手段であって、前記第1のサブモデルおよび前記第2のサブモデルが前記第1のカプラー、前記バッファ、および前記第2のカプラーを用いて通信する、手段とを備えた手段を備えた方法。

【請求項17】前記モデルが論理設計である、請求項16に記載のシステム。

【請求項18】前記一括して処理する手段が前記第1のサブモデルおよび前記第2のサブモデルを分散処理する手段を備え、

前記分散処理する手段が分散プロセス制御を備えている、請求項16に記載のシステム。

【請求項19】前記一括して処理する手段がさらに前記第1のサブモデルを第1のシミュレータ・インスタンス

上で処理する手段と前記第2のサブモデルを第2のシミュレータ・インスタンス上で処理する手段とを備え、前記第1のシミュレータ・インスタンスは前記第2のシミュレータ・インスタンスと異なり、

前記第1のシミュレータ・インスタンスは前記バッファを用いて前記第2のシミュレータ・インスタンスと通信する、請求項16に記載のシステム。

【請求項20】前記第1のシミュレータ・インスタンスは第1のコンピューティング・ユニットに常駐し、前記第2のシミュレータ・インスタンスは第2のコンピューティング・ユニットに常駐する、請求項19に記載のシステム。

【請求項21】少なくとも1つのサブモデルおよび付随するカプラーが少なくとも1つのマップされたデータ・ポートおよびクロック/サイクル・ポートを備え、前記クロック/サイクル・ポートは少なくとも1つのクロック/サイクル信号を供給し、

前記少なくとも1つのクロック/サイクル信号は前記少なくとも1つのサブモデルと付随するカプラーとの間で前記少なくとも1つのマップされたデータ・ポートを通じてデータを転送するのに使用される、請求項16に記載のシステム。

【請求項22】前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方がトップ・レベル・カプラー・エンティティとカプラー共有ライブラリとを備え、

前記第1のカプラーおよび前記第2のカプラーのうちの前記少なくとも一方が前記トップ・レベル・カプラー・エンティティを用いて前記第1のサブモデルおよび前記第2のサブモデルと個別に接続し、

前記トップ・レベル・カプラー・エンティティが少なくとも1つの入力と、少なくとも1つの出力と、前記第1のサブモデルおよび前記第2のサブモデルのうちの少なくとも一方の前記クロック/サイクル・ポートにマップされた複数のポートとを備え、

前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方が前記カプラー共有ライブラリを用いて前記カプラーと前記バッファとの間でデータを転送する、請求項21に記載のシステム。

【請求項23】前記バッファが通信媒体を備え、前記通信媒体は前記第1のカプラーと前記第2のカプラーとの間でデータを渡すとともに監視する少なくとも1つのプロトコルを備えている、請求項16に記載のシステム。

【請求項24】前記一括して処理する手段がさらに、前記第1のカプラーが前記第1のサブモデルから取得したデータを前記バッファに送信し、前記第2のカプラーが前記データを前記バッファから取得し、前記データを前記第2のサブモデルに供給する手段と、

前記送信と前記取得とを対応させることにより、前記デ

ータの一貫性を保証する手段とを備えた、請求項 16 に記載のシステム。

【請求項 25】前記一貫性を保証する手段が、前記データにチェックサム値を挿入する手段と、前記第 1 のカプラーおよび前記第 2 のカプラーのうちの一方が前記バッファ中の前記データを調べて前記データの正確性を判断する手段と、前記データが不正確な場合、前記第 1 のカプラーおよび前記第 2 のカプラーのうちの少なくとも一方が、前記データの

前記バッファへの再送信および前記データの再読み取りのうちの少なくとも一方を行なう手段とを備えた、請求項 24 に記載の方法。

【請求項 26】前記挿入する手段が、前記チェックサム値を計算する手段を備え、前記計算する手段が、少なくとも 1 つの論理操作を前記データに適用する手段を備えている、請求項 25 に記載のシステム。

【請求項 27】前記調べる手段が、前記データの新たなチェックサム値を計算する手段と、前記新たなチェックサム値と前記チェックサム値とを比較する手段とを備えた、請求項 25 に記載のシステム。

【請求項 28】前記一貫性を保証する手段がさらに、前記バッファに、前記第 1 のカプラーが書き込みアクセス可能であるとともに前記第 2 のカプラーが読み取りアクセス可能な第 1 の一連ファイルを準備する手段と、前記バッファに、前記第 1 のカプラーが読み取りアクセス可能であるとともに前記第 2 のカプラーが書き込みアクセス可能な第 2 の一連ファイルを準備する手段と、所定の順序で、前記第 1 のカプラーが前記第 1 の一連ファイルに書き込み、前記第 2 のカプラーが前記第 1 の一連ファイルから読み取る手段と、前記第 2 のカプラーが前記第 2 の一連ファイルに書き込み、前記第 1 のカプラーが前記第 2 の一連ファイルから読み取る手段とを備えた、請求項 24 に記載のシステム。

【請求項 29】前記インタフェースさせる手段がさらに、前記バッファ中のデータを監視して監視統計を供給する手段を備え、前記監視統計が前記第 1 のサブモデルおよび前記第 2 のサブモデルを統計的に再分割して前記一括処理を最適化するのを容易にする、請求項 16 に記載のシステム。

【請求項 30】前記モデルが VHD L で表されている、請求項 16 に記載のシステム。

【請求項 31】分割されたモデルの、第 1 のサブモデルおよび第 2 のサブモデルを含む複数のサブモデルを処理するシミュレーション・システムであって、分割されたモデルの第 1 のサブモデルと第 2 のサブモデルとを、前記第 1 のサブモデルにインタフェースした第 1 のカプラーおよび前記第 2 のサブモデルにインタフェースした第 2 のカプラーを準備し、前記第 1 のカプラー

と前記第 2 のカプラーとの間に設けたバッファによって前記第 1 のカプラーと前記第 2 のカプラーとをインタフェースさせることにより、結合するように適合した少なくとも 1 つのコンピューティング・ユニットを備え、前記少なくとも 1 つのコンピューティング・ユニットがさらに前記第 1 のサブモデルおよび前記第 2 のサブモデルを一括して処理するように適合しており、前記第 1 のサブモデルおよび前記第 2 のサブモデルが前記第 1 のカプラー、前記第 2 のカプラー、および前記第 2 のカプラーを用いて通信するシミュレーション・システム。

【請求項 32】前記少なくとも 1 つのコンピューティング・ユニットが複数のシミュレータ・インスタンスを備え、前記複数のシミュレータ・インスタンスが第 1 のシミュレータ・インスタンスおよび第 2 のシミュレータ・インスタンスを含み、前記第 1 のシミュレータ・インスタンスが前記第 1 のサブモデルを処理し、前記第 2 のシミュレータ・インスタンスが前記第 2 のサブモデルを処理し、前記第 1 のシミュレータ・インスタンスが前記第 2 のシミュレータ・インスタンスとは異なる、請求項 31 に記載のシミュレーション・システム。

【請求項 33】前記第 1 のシミュレータ・インスタンスが第 1 のコンピューティング・ユニットに常駐し、前記第 2 のシミュレータ・インスタンスが第 2 のコンピューティング・ユニットに常駐する、請求項 32 に記載のシミュレーション・システム。

【請求項 34】分割されたモデルの、第 1 のサブモデルおよび第 2 のサブモデルを含む複数のサブモデルを処理する方法であって、分割されたモデルの第 1 のサブモデルと第 2 のサブモデルとを結合するステップであって、前記第 1 のサブモデルがインタフェースする第 1 のカプラーを準備するステップと、前記第 2 のサブモデルがインタフェースする第 2 のカプラーを準備するステップと、前記第 1 のカプラーと前記第 2 のカプラーとをインタフェースさせるステップであって、前記第 1 のカプラーと前記第 2 のカプラーとの間にバッファを準備するステップとを備えたステップと、前記第 1 のサブモデルおよび前記第 2 のサブモデルを一括して処理するステップであって、前記第 1 のサブモデルおよび前記第 2 のサブモデルが前記第 1 のカプラー、前記第 2 のカプラーを用いて通信する、ステップとを備えたステップを備えた方法を実行する、機械によって実行可能な命令から成る少なくとも 1 つのプログラムを有形的に記録した、機械読み取り可能なプログラム記憶装置。

【請求項 35】前記モデルが論理設計である、請求項 34 に記載のプログラム記憶装置。

【請求項36】前記一括して処理するステップが前記第1のサブモデルおよび前記第2のサブモデルを分散処理するステップを備え、

前記分散処理するステップが分散プロセス制御を備えている、請求項34に記載のプログラム記憶装置。

【請求項37】前記一括して処理するステップがさらに前記第1のサブモデルを第1のシミュレータ・インスタンス上で処理するステップと前記第2のサブモデルを第2のシミュレータ・インスタンス上で処理するステップとを備え、

前記第1のシミュレータ・インスタンスは前記第2のシミュレータ・インスタンスと異なり、

前記第1のシミュレータ・インスタンスは前記バッファを用いて前記第2のシミュレータ・インスタンスと通信する、請求項34に記載のプログラム記憶装置。

【請求項38】前記第1のシミュレータ・インスタンスは第1のコンピューティング・ユニットに常駐し、前記第2のシミュレータ・インスタンスは第2のコンピューティング・ユニットに常駐する、請求項37に記載のプログラム記憶装置。

【請求項39】少なくとも1つのサブモデルおよび付随するカプラーが少なくとも1つのマップされたデータ・ポートおよびクロック／サイクル・ポートを備え、前記クロック／サイクル・ポートは少なくとも1つのクロック／サイクル信号を供給し、前記少なくとも1つのクロック／サイクル信号は前記少なくとも1つのサブモデルと付随するカプラーとの間で前記少なくとも1つのマップされたデータ・ポートを通じてデータを転送するのに使用される、請求項34に記載のプログラム記憶装置。

【請求項40】前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方がトップ・レベル・カプラー・エンティティとカプラー共有ライブラリとを備え、

前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方が前記トップ・レベル・カプラー・エンティティを用いて前記第1のサブモデルおよび前記第2のサブモデルと個別に接続し、

前記トップ・レベル・カプラー・エンティティが少なくとも1つの入力と、少なくとも1つの出力と、前記第1のサブモデルおよび前記第2のサブモデルのうちの少なくとも一方の前記クロック／サイクル・ポートにマップされた複数のポートとを備え、

前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方が前記カプラー共有ライブラリを用いて前記カプラーと前記バッファとの間でデータを転送する、請求項39に記載のプログラム記憶装置。

【請求項41】前記バッファが通信媒体を備え、前記通信媒体は前記第1のカプラーと前記第2のカプラーとの間でデータを渡すとともに監視する少なくとも1

つのプロトコルを備えている、請求項34に記載のプログラム記憶装置。

【請求項42】前記一括して処理するステップがさらに、

前記第1のカプラーが前記第1のサブモデルから取得したデータを前記バッファに送信し、前記第2のカプラーが前記データを前記バッファから取得し、前記データを前記第2のサブモデルに供給するステップと、

前記送信と前記取得とを対応させることにより、前記データの一貫性を保証するステップとを備えた、請求項34に記載のプログラム記憶装置。

【請求項43】前記一貫性を保証するステップが、前記データにチェックサム値を挿入するステップと、前記第1のカプラーおよび前記第2のカプラーのうちの一方が前記バッファ中の前記データを調べて前記データの正確性を判断するステップと、

前記データが不正確な場合、前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方が、前記データの前記バッファへの再送信および前記データの前記バッファからの再読み取りのうちの少なくとも一方を行なうステップとを備えた、請求項42に記載のプログラム記憶装置。

【請求項44】前記挿入するステップが、前記チェックサム値を計算するステップを備え、前記計算するステップが、少なくとも1つの論理操作を前記データに適用するステップを備えている、請求項43に記載のプログラム記憶装置。

【請求項45】前記調べるステップが、前記データの新たなチェックサム値を計算するステップと、前記新たなチェックサム値と前記チェックサム値とを比較するステップとを備えた、請求項43に記載のプログラム記憶装置。

【請求項46】前記一貫性を保証するステップがさらに、

前記バッファに、前記第1のカプラーが書き込みアクセス可能であるとともに前記第2のカプラーが読み取りアクセス可能な第1の一連ファイルを準備するステップと、

前記バッファに、前記第1のカプラーが読み取りアクセス可能であるとともに前記第2のカプラーが書き込みアクセス可能な第2の一連ファイルを準備するステップと、

所定の順序で、前記第1のカプラーが前記第1の一連ファイルに書き込み、前記第2のカプラーが前記第1の一連ファイルから読み取るステップと、

前記所定の順序で、前記第2のカプラーが前記第2の一連ファイルに書き込み、前記第1のカプラーが前記第2の一連ファイルから読み取るステップとを備えた、請求

項42に記載のプログラム記憶装置。

【請求項47】前記インタフェースさせるステップがさらに、前記バッファ中のデータを監視して監視統計を供給するステップを備え、

前記監視統計が前記第1のサブモデルおよび前記第2のサブモデルを統計的に再分割して前記一括処理を最適化するのを容易にする、請求項34に記載のプログラム記憶装置。

【請求項48】前記モデルがVHDLで表されている、請求項34に記載のプログラム記憶装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、一般にモデル・シミュレーションに関し、特に分割されたモデルのサブモデル群をコンカレント分散サブモデル・シミュレーション実行用に緊密かつ機能的に結合する手法に関する。

【0002】

【従来の技術】モデル・シミュレーションは実システムを設計しその性能を予測する際における相変わらず重要な「ツール」である。モデル・シミュレーションは新たなシステムの適用による性能予測を実装前に生成するのにも役立つ。間違ったシステムを設計する危険性がきわめて大きい複雑なシステムでは、モデル・シミュレーションは必須である。

【0003】分散シミュレーションとは、手元のシステムを様々なコンピュータ・ワークステーションに割り当てられ構成要素をなすサブモデルに分割して実行するシミュレーション環境のことである。

【0004】一般に、分散シミュレーションでは、システム・サブモデル群を実行している協働ワークステーション群は最小限、各ワークステーションが実行すべき次のイベントに加え当該実行用に予定された時間を指示する、タイムスタンプを押されたイベント情報を交換する必要がある。従来技術では、分散シミュレーション用に2種類のモデルを用いている。すなわち、(a)「オプティミスティック(楽観的)」モデルと(b)「コンサーバティブ(保守的)」モデルである。

【0005】オプティミスティック・モデルでは、様々なプロセッサが処理するイベントの実行を順序付けない。その代わり、オプティミスティック・モデルでは、各ワークステーションは自分自身のイベント・シーケンスを別のワークステーションが処理するイベントとは無関係であるとみなして実行することができる。同時に、オプティミスティック・モデルには、因果関係エラーの検出とロールバックによって行なう引き続く復旧のための機構が実装されている。

【0006】他方、コンサーバティブ・モデルは様々なワークステーションが実行するイベント間の相互依存関係を特定する先読みアルゴリズムを実装することにより、因果関係エラーを完全に排除するという観点に立つ

ている。したがって、コンサーバティブ・モデルでは、あるワークステーションがイベントを処理できるのは、問題のイベントが、並行して処理されているイベントまたは他のワークステーションが次に処理するはずのイベントの結果に影響されないことが分かっている場合のみである。

【0007】オプティミスティック手法の欠点のうちで最も顕著なものは、必要とされる検出・ロールバック機構がきわめて複雑であり実装するのが困難であるという事実である。

【0008】コンサーバティブ・モデルは、上述したオーバーヘッドおよび不安定性とは無縁であるけれども、良好な性能を達成するためにはイベントの並列性を特定して利用する効率的で複雑な先読みアルゴリズムを必要とする。多くのアプリケーションがそのように効率的なアルゴリズムを開発できていない点を考えると、これは深刻な問題である。コンサーバティブ・モデルに付随する別の問題は、ユーザはシミュレートしている特定のアプリケーション/実験用のモデルを「微調整」できるように、使われているイベント同期方式を詳しく知っている必要があるという事実に関わっている。

【0009】上述した2種類のモデルに共通する別の欠点は、(既製の市販ソフトウェアと異なり)特定の研究用または特定のアプリケーション・プログラム用に開発された特別のソフトウェアに依存している点である。さらに別の共通の欠点は、すべてのワークステーションが同じシミュレータのインスタンスを実行する必要があるという点である。

【0010】上述した観点から、同じまたは異なるシミュレータのインスタンスを実行しているワークステーション群が通信するのを可能にするとともに、単純な検出・ロールバック機構を実現できる一般的な機構が当技術分野で求められている。

【0011】

【課題を解決するための手段】本発明は一側面において、分割されたモデルの複数のサブモデルを処理する方法を提供することにより、従来技術の欠点を克服しさらなる利点を提供するものである。この方法はたとえば各々が対応するサブモデルにインタフェースした複数のカプラーを用いて上記複数のサブモデルを結合するステップと、上記複数のサブモデルを一括して処理するステップとを備え、上記複数のサブモデルが上記複数のカプラーおよび共通バッファを用いて通信するものである。

【0012】本発明の別の側面では、上記複数のサブモデルを一括して処理するステップが、様々なシミュレータの複数のインスタンスを用いシミュレーション・プロセスの目標に従って各サブモデルを個別に処理しうるようにするステップを備えている。

【0013】本発明のさらに別の側面では、上記バッファ中のデータが正確であるか否かを自動的に調べ、必要

な場合には再送信する。

【0014】ここでは、上述した方法に対応するシステムとコンピュータ・プログラム製品についても述べられている。

【0015】本発明の諸側面によれば、共有カプラー・ライブラリとライセンスを受けたハードウェア・シミュレータのAPI（アプリケーション・プログラミング・インタフェース）とを用いることにより、ライセンスを受けたハードウェア・シミュレータのソース・コードにわざわざアクセスしなくともモデルの分散処理を効果的に行なえる。本発明によれば、サブモデルが互いに直接に通信しうるようにすることにより、ハブレス（中心なし）分散処理、すなわち分散制御による処理が可能になる。さらに、ここに提示するモデル処理手法はコンピューティング・ユニット、そのアーキテクチャ、そのオペレーティング・システム、およびライセンスを受けたハードウェア・シミュレータで使われるプログラミング言語に関し特定の要件すなわち限定を必要としない。これは、共有カプラー・ライブラリの汎用機能を用いライセンスを受けたハードウェア・シミュレータのAPIと直接に通信することにより、かつ、様々なアーキテクチャおよび/またはオペレーティング・システムの特定の機能に依存しないことにより実現されている。（「Aおよび/またはB」は「AおよびB、A、またはB」を表わす。）

【0016】本発明の手法によれば、さらなる機能と利点を実現しうる。本発明の他の実例と側面をここで詳細に説明するが、それらは特許請求の範囲に記載した本発明の一部と考えられる。

【0017】

【発明の実施の形態】本発明の少なくとも1つの側面により、複数のサブモデルに分割された1つのモデルを一括して処理する手法を提示する。この手法によれば、たとえば同じライセンスを受けたハードウェア・シミュレータまたは異なるライセンスを受けたハードウェア・シミュレータまたは複数のサブモデルを処理するライセンスを受けたプログラムのインスタンス群を接続する汎用通信機構を備えることによりモデルを処理するのが容易になる。

【0018】以下、本発明の諸側面を取り込んで使用している、分散シミュレーション・システムの一実施形態を図1を参照して説明する。この分散コンピューティング環境はライセンスを受けたハードウェア・シミュレータのインスタンス104を実行するコンピューティング・ユニット100と、ライセンスを受けたハードウェア・シミュレータのインスタンス106を実行するコンピューティング・ユニット102とを備えている。各コンピューティング・ユニットは、たとえばインターナショナル・ビジネス・マシーンズ・コーポレーション（IBM）が提供するUNIX(R)ベースのオペレーティング

・システムであるAIXを実行するIBM RISC/6000コンピュータである。一実施形態では、インスタンス104、106は異なるライセンスを受けたハードウェア・シミュレータ、たとえばモデル・テクノロジー社が提供するVSIMやインターナショナル・ビジネス・マシーンズ・コーポレーションが提供するPSIMなどのインスタンスである。しかし、別の実施形態では、インスタンス104、106は同じライセンスを受けたハードウェア・シミュレータのインスタンスであってもよい。

【0019】インスタンス104はカプラー110がインタフェースしているサブモデル108を処理する。

（「インタフェースする」とはある構成要素が異質の他の構成要素に接続することである。）一方、インスタンス106はカプラー114がインタフェースしているサブモデル112を処理する。サブモデル108、112は処理すべきハードウェアのモデルを分割することにより生成する。様々な分割手法のさらなる詳細は文献や本願の同時米国出願に述べられている。ハードウェア・モデル分割の一例を下で図2（a）（b）（c）を参照して説明する。一方、本発明の一側面に係る汎用カプラーの一実施形態、および当該カプラーでサブモデルにインタフェースする方法を下で図3を参照して説明する。

【0020】図1の参照を続ける。カプラー110、114にはバッファ116がインタフェースしている。バッファ116は一般に、カプラー110、114がアクセス可能でサブモデル108、112が通信するのを可能にする通信媒体を備えている。一例を挙げると、この通信媒体にはIBMが提供しているAFS（Andrew File System）を用いた記憶装置を用いることができる。AFSとは、協働しているホスト群がLAN（local area network）とWAN（wide area network）の双方をまたいでファイルシステム資源を効率的に共有するのを可能にする分散ファイルシステムのことである。この特徴により、本発明でAFSを使用することができる。さらに、上記記憶装置には共通通信ディレクトリ（CCD）を備えることができる。このCCDはカプラー110、114が読み書きアクセス可能な複数ファイル118を含んでいる。一実施形態では、この複数ファイルはモデルまたは分割されたサブモデルの処理が始まる前にCCD中に存在している。複数ファイル118はサブモデル108と112の間でカプラー110と114を介してデータを転送するのに使うことができる。一例では、カプラー110がサブモデル108からデータを取得し、それを複数ファイル118のうちの少なくとも1つのファイルに書き込む。次いで、カプラー114がこのファイルからデータを読み取り、当該データをサブモデル112に渡す。サブモデル112からサブモデル108へのデータの転送も同様の方法で行なう。一例として、ファイルにデータを書き込むのに用いる論理の一実施形態



を下で図7を参照して詳述し、ファイルからデータを読み取るのに用いる論理の一実施形態を下で図8を参照して詳述する。

【0021】一実施形態では、複数ファイル118は2つのファイルから構成することができる。この実施形態では、一方のファイルはサブモデル108からサブモデル112にデータを転送するのに使用することができ、他方のファイルはサブモデル112からサブモデル108にデータを転送するのに使用することができる。

【0022】別の実施形態では、入出力速度を速めるために、複数のサブモデル間でデータを転送するのに一連ファイルを用いることができる。一例として、一連ファイルに所定の順序でデータを書き込むことができる。次いで、このデータは当該一連ファイルから同じ所定の順序で読み取ることができる。したがって、書き込みの間の間隔が保証されているから、データをバッファに書き込もうとしているカプラーは当該バッファ中のファイルが上書きされないかどうかを調べる必要がない。一例では、一連ファイルは決定論的なラウンド・ロビン方式で読み書きされる。任意のカプラーが任意のファイルを再書き込みしうようになるには、その前にサブモデルの一括処理全体がデータの交換を多数回結合することにより先行している必要がある。これにより、より速く処理されるサブモデルが待機することなく少なくともいくつかのデータを出力しうることが保証される。同様に入力側では、より遅く処理されるサブモデルが、読み取る入力データを取得することが保証される。

【0023】上述したシステムとコンピューティング・ユニットは例として提示しただけである。本発明の通信インタフェースは、本発明の本旨のうちで、様々な種類のコンピューティング・ユニット、コンピュータ、プロセッサ、ノード、システム、ワークステーション、および/または環境に内蔵する、あるいはそれらとともに使用することができる。たとえば、コンピューティング・ユニットのうちの少なくとも1つはUNIX(R)アーキテクチャに基づいていてもよいし、あるいはインテルPCアーキテクチャを内蔵していてもよい。さらに、ここで説明するいくつかの実施形態は2つのサブモデルを処理する2つのコンピューティング・ユニットしか備えていないが、任意個数のサブモデルを処理する任意個数のコンピューティング・ユニットを用いることができる。さらに、通信媒体としては、ここで説明する通信プロトコルを順守する任意の型の通信媒体、たとえばパケットまたはメッセージの送受信を介したネットワークなどを用いることができる。また、他の型のシステムも本発明の利益を享受できるから、本発明の一部をなすと考えられる。さらに、上述した方法には、オペレーティング・システム、あるいは、ライセンスを受けたハードウェア・シミュレータもしくはインスタンス中のライセンスを受けたプログラムで使用されている、またはそれに潜在

するプログラミング言語に対する特定の要件は付随しない(すなわちオペレーティング・システムまたはプログラミング言語として任意のものを使用することができる)。

【0024】以下、上述したように、本発明の機能を用いてシミュレーションを実行するあるモデルの一実施形態を図2(a)(b)(c)を参照して説明する。未分割のモデル200は、たとえば動作コンポーネント202とクロック/サイクル・エンティティ204を備えている。クロック/サイクル・エンティティ204は動作コンポーネント202が変化しうる時を決める。

【0025】ASIC(特定用途向け集積回路)のチップ設計概念では、モデルの動作コンポーネントはラッチ、ゲート、および配線を備えている。一方、変化画定エンティティはクロックが分布した配線上のクロック波形値である。動作コンポーネントは新たな値を取得し、クロック波形のサイクルに基づいて新たな値を出力する。イベント・ドリブン用途ではクロック・イベント、たとえば波形の立ち上がりを利用する。一方、サイクル・ドリブン用途ではクロック/サイクル遷移を利用する。上記2種類の用途はそれぞれのエンティティを用いてモデルの動作コンポーネントが変化しうる時を決める。

【0026】一実施形態では、モデルは、たとえば図2(b)に詳細に示すサブモデル206、208のように、内部で互いに通信している複数のサブモデルを備えているように概念的に説明することができる。サブモデル206、208の動作コンポーネントは入出力218を備えているように表わすことができる。入出力218はサブモデル間のデータ交換用に使われる。クロック/サイクル・エンティティ214は両サブモデルに共通であるが、サブモデル206と208の間に分布しており、チャンネル216を介して結合されているように表わすことができる。モデルはサブモデルに分割することができる。それには、サブモデル206の入力をサブモデル208の出力から分離するとともにサブモデル208の入力をサブモデル206の出力から分離する。そして、サブモデル間にまたがるクロック/サイクル・エンティティ214をチャンネル216のところで分割する。

【0027】モデル200のサブモデルへの概念的な分割を図2(c)に示す。分割の目的は各々が入力222、224と出力220、226を備えたサブモデルを生成することであり、分割に由来するすべてのサブモデルに共通のクロック/サイクル・エンティティを実現することでもある。実現されたクロック/サイクル・エンティティは各サブモデルにおいてクロック/サイクル・ポート228、230として表わすことができる。

【0028】実際の分割は良く定義されたインタフェースを目標とする人手によるプロセス、またはインタフェースを構築する際に依拠するアルゴリズム・プロセスに

なる可能性がある。上述した同時特許出願には、自動分割プロセスの例が記載されている。

【0029】本発明の一側面として、ここでは、各サブモデルにインタフェースしてたとえば異なるコンピューティング・ユニットで実行されているサブモデル間の通信を可能にする汎用カプラーを提示する。そのようなカプラーの一例、およびインタフェース動作の一例を図3に示す。

【0030】一実施形態として、サブモデル300をVHDL (VHSIC (Very High Speed Integrated Circuits) Hardware Description Language) を用いて表わす。VHDLはIEEE (Institute of Electrical and Electronics Engineers)が開発した標準 (VHDL-1076)である。VHDLにおけるサブモデルは、ここで用いているように、少なくとも1つの出力310、少なくとも1つ入力312、およびクロック/サイクル・ポート314を備えたエンティティである。この実施形態では、カプラー302はトップ・レベル・カプラー・エンティティ304を備えている。そして、トップ・レベル・カプラー・エンティティ304自体がVHDLトップ・レベル・エンティティと汎用VHDLカプラー・エンティティを備えている。一接続形式では、VHDLトップ・レベル・エンティティを用いてサブモデルの入力、出力、およびクロック/サイクル・ポートを汎用VHDLカプラー・エンティティの標準ポートに接続することができる。標準ポートには出力ポート316、入力ポート318、およびクロック/サイクル・ポート320がある。

【0031】VHDLトップ・レベル・エンティティ内では、サブモデルと汎用VHDLカプラー・エンティティをインスタンス化したのち、ポート・マップを介してトップ・レベルの信号に接続する。サブモデル300の一例を図4に示す。汎用VHDLカプラー・エンティティの一例を図5に示す。そして、VHDLトップ・レベル・エンティティ、およびトップ・レベル・カプラー・エンティティ304とサブモデル300との間の接続形式を図6に示す。

【0032】図3の参照を続ける。トップ・レベル・カプラー・エンティティ304の入力ポート318と出力ポート316はサブモデル300の入力312と出力310にそれぞれマップされている。これにより、カプラー302とサブモデル300との間の通信が可能になる。クロック/サイクル・ポート320はサブモデル300のクロック/サイクル・ポート314にマップされている。これにより、カプラー302がサブモデル300のクロック/サイクル・エンティティを検出するのが可能になる。カプラー302はクロック/サイクル・ポート314の信号を利用して、出力310から別のサブモデルへ送信するデータを集めたり、CCDから入力312へデータを渡したりする。さらに、カプラー302はカプラー共有ライブラリ308を備えている。カプラ

ー共有ライブラリ308には、サブモデルが入出力するデータ・フローをカプラー302が制御するのを可能にする機能が含まれている。

【0033】一実施形態では、クロック/サイクル・ポート314で変化が検出されると、カプラー共有ライブラリ308が呼び出される。カプラー302はカプラー共有ライブラリ308が備えている機能 (図7〜図10参照) を使って、サブモデルから出力値を集めその情報を出力送信先たとえばCCDに渡す、あるいは、入力源たとえばCCDから入力情報を集めその情報をサブモデルに渡す。

【0034】一例では、カプラー共有ライブラリはCプログラミング言語を用いて実装することができる。この実施形態では、カプラー302はさらに、トップ・レベル・カプラー・エンティティとカプラー共有ライブラリをインタフェースする、VHDLへのAPI (application programming interface) 306を備えている。

【0035】以下、カプラー共有ライブラリが備え、カプラーがCCDからデータを入力するとき、およびCCDへデータを出力するときに使ういくつかの論理を図7〜図10を参照して詳細に説明する。

【0036】あるサブモデルから別のサブモデルにデータを送信するには、送信すべきデータをまずCCDに出力する。一実施形態では、このデータを図7に示す論理に従ってファイルに書き込む。

【0037】まず、カプラーが書き込むべきデータを検索・取得、すなわちサブモデルからデータを取得する (500)。一実施形態では、カプラーがこのデータのコピーをローカル記憶装置 (図示せず) に格納する。この格納したデータは再送信機能を実行する次のサイクルで必要になる可能性がある。次いで、カプラーは書き込み用のあるファイルを読み取り用にオープンし (502)、当該ファイルを読み取って再送信要求がないかどうか各レコードを調べる (504)。再送信要求があるとしたら、それは上記ファイル中のデータが正しくないことを見いだした先行読み取り者が上記ファイルに書き込んだものである。ところで、高信頼性環境では、データを再送信する機能はシステム全体を高速化するために任意実行事項として無効にうる。再送信要求をファイルに書き込むのに使う論理の例を下で図10を参照して詳述する。

【0038】図7の参照を続ける。再送信要求が存在する場合 (506)、カプラーは再送信が有効であるか否かを調べる (520)。Yesの場合、ファイルをクローズし、カプラーは再送信機能呼び出す (526)。再送信機能の一実施形態は下で図8を参照して詳述する。再送信機能は先行サイクルでローカル記憶装置に格納したデータをカプラーに供給し、ファイルへのデータの書き込みを直ちに継続する (514)。データの再送信が有効でない場合、ファイルをクローズし、カプラー

は書き込み手順を終了する(524)。一実施形態では、発生した問題点を記載したログ・ファイルを生成してもよい。

【0039】再送信要求が存在しない場合、カプラーはレコードにINVALID(無効)キャラクタがないかどうか調べる(508)。すべてのレコードにこのキャラクタが存在しない場合、このファイルは最後まで読み取られていないから、上書きしてはいけない。この場合、カプラーはファイルをクローズし(510)、書き込み手順を初めから開始する(502)。すべてのレコードがINVALIDとマークされている場合には、当該ファイルは上書きできる状態にある。カプラーは読み取り用の当該ファイルをクローズしたのち、それを書き込み用にオープンする(512)。

【0040】次に、カプラーはデータを当該ファイルに書き込む(514)。一実施形態では、カプラーは当該ファイルの各レコードにVALID(有効)キャラクタ、順序レコード番号、正確かつ同一のサイクル番号、およびレコード固有データを書き込む。サイクル番号はデータに付随するクロック・サイクルを特定するものである。順序レコード番号は一実施形態では、当該ファイル中の各レコードの序数を表しており、エラー検出用に使うことができる。チェックサムを計算し、当該ファイルの最終レコードにレコード固有データとして書き込むのが望ましい(516)。一実施形態では、チェックサムの計算は当該ファイルのすべてのレコードに排他的OR操作を16進形式で適用することにより行なっている。このあと、当該ファイルをクローズした(518)のち、書き込み手順は終了する(528)。

【0041】以下、カプラーが再送信機能を実行するのに使用する論理の一実施形態を図8を参照して説明する。

【0042】まず、カプラーは書き込み用にファイルをオープンする(602)。次いで、カプラーは先行サイクルで格納されたデータを取得し(604)、それを上記ファイルに書き込む準備をする(606)。このあと、カプラーは書き込み手順を開始する。

【0043】データ再送信の第2の部分はCCDからデータを検索・取得して、付随するカプラーがサブモデルに入力するのに備えることである。一実施形態では、このデータはたとえば図9に示す論理に従ってファイルから読み取る。

【0044】まず、カプラーがクロック/サイクル・ポートを介してそのサブモデルがデータを必要としていることを検出したのち、読み取り手順を開始する(700)。カプラーは上記サブモデルに入力すべきデータを格納しているCCD中のファイルを読み取り用としてオープンする(702)。次いで、カプラーが当該データを上記ファイルからロードする(704)。

【0045】次に、VALIDキャラクタ、正確な順序

レコード番号、および正確かつ同一のサイクル番号をあるかどうか上記ファイルのレコード群を調べる(708)。あるサブモデルが他のサブモデルよりも速く処理される可能性があるから、上記ファイル中のデータが読み取りに適した状態にないことがありうる。もしも、あるレコード中に不正確なデータが存在する場合には、当該ファイルをクローズした(710)のち、読み取り手順を繰り返す(702)。

【0046】すべてのレコードが読み取りに適した状態にあるとともに正確なデータを含んでいる場合、カプラーは当該すべてのレコードに対して16進形式で累積的排他ORを計算することによりチェックサム値を算出する(712)。ただし、最終レコードはチェックサム値を格納しているから、この計算から除外する。次いで、算出したチェックサム値と最終レコードに格納されているチェックサム値とを比較する(714)。

【0047】2つのチェックサム値が一致しない場合、カプラーはエラーを報告した(716)のち、再送信が有効であるか否かを調べる(718)。Yesの場合、当該読み取り用ファイルをクローズした(720)のち、カプラーは再送信要求手順を開始する(722)。再送信要求機能で使われる論理の一例を下で図10を参照して説明する。再送信を要求したのち、読み取り手順を繰り返す(702)。読み取り手順は次の書き込み者が再送信を実行するまで繰り返すことになる。再送信が有効でない場合には、カプラーは当該ファイルをクローズしたのち、デバッグ・ログ機能を実行する(724)。

【0048】2つのチェックサム値が一致する場合、カプラーは次のサイクルのために最終チェックサム値の格納を開始する。次いで、当該ファイルを読み取り用にはクローズしたのち、書き込み用にオープンして(730)、当該ファイルの将来の読み取り者に当該ファイルがすでに読み取られていることを知らせることができるようにする。そうするために、カプラーは各レコードにINVALID(無効)とマーク付ける。次いで、当該ファイルをクローズした(734)のち、カプラーは読み取り手順を終了する(736)。この時点で、カプラーは当該ファイルから読み取ったデータを保持しているから、それをサブモデルに当該サブモデルの入力を介して渡すことができる。

【0049】以下、カプラーが再送信を要求するのに使用しうる論理の一例を図10を参照して説明する。

【0050】カプラーが読み取りプロセスの間にチェックサム・エラーに遭遇すると、図10の手順が開始する(800)。まず、カプラーはファイルを書き込み用にオープンし(802)、当該ファイルの各レコードに再送信要求を書き込んだ(804)のち、当該ファイルをクローズする(806)。これにより、次の書き込み者は再送信機能(図7、図8に示したように)を実行すべ

きことを知らされることになる。以後、カプラーは読み取り手順を開始する。

【0051】一実施形態では、処理システムの信頼性が十分に高い場合には、当業者にとって明らかなように、ファイルの一貫性を保証する論理を上述したプロセス・フローから取り除いてシステム全体の速度を高めることができる。そのような実施形態では、書き込み手順は検査プロセスを含んでおらず、読み取り手順はサイクル番号しか検査しない。速く処理されるサブモデルは遅く処理されるサブモデルを待つとともに、集合体の同期性を維持するためにファイルを再読み取りすることを強制される。これに対して、最も遅いサブモデルはつねに入力に適した状態の入力データを取得するとともに、再読み取りは全く行なわない。したがって、再読み取り回数は処理コンピューティング・ユニットに課される負荷の尺度である。たとえば、再読み取り回数が少ないということは負荷が大きい、すなわちコンピューティング・ユニットが遅いということを表わしている。

【0052】一実施形態では、シミュレーション実行間でサブモデルを静的に再分割するのに再読み取り回数の統計を利用することができる。別の実施形態では、たとえば遅いアプリケーション・インスタンスの優先順位を引き上げることにより、コンピューティング資源をこの遅いアプリケーション・インスタンスに動的に割り当てるのに再読み取り回数の統計を利用することができる。この実施形態では、1つの遅いポイント（遅いインスタンス）をインスタンスの集合と取り替えても、モデル集合体としては計算が進行する速度が低下することはない。さらに別の実施形態では、ある集合体中のすべてのコンピューティング・ユニットにわたる再読み取り回数の統計は、当該集合体全体の性能の尺度として利用することができる。

【0053】さらに別の実施形態では、ファイルに書き込むのに適した状態にあるデータを連結して1つのレコードにすることができるから、当該データを出力するのにたった1回の書き込みしか必要としない。同様に、ファイルから読み取るのに適した状態にあるデータを当該ファイルから入力するには、たった1回の読み取りしか必要としない。たった1回の読み取りでファイルから取得したデータは、入力側で分解することができる。

【0054】さらに別の実施形態では、サブモデルを処理するハードウェア・シミュレータまたはアプリケーション・プログラムの各インスタンスを一連のコンピュータにわたり非同期に開始することができる。各インスタンスが先に進むのを止められるのは、入力資源または出力資源がそれを強制的に待たせる場合のみである。このように、インスタンス群は全体として非接続で（すなわち互いに無関係に）開始することができ、集合体中の他の構成要素がCCD中のファイルを更新するまでずっと先に進むことができる。同様に、集合体全体を所定の時

点まで進ませる、あるいは所定の時点で停止させるのに、キーボード入力を用いたフォアグラウンド（前景）セッションを利用することができる。

【0055】ここで開示した結合機構によって、最も遅いサブモデルのインスタンスは、より速いインスタンスから必要な情報を取得するのに自身が費やす時間を最小限にすることにより、可能な限り最大の速度で実行することが可能になる。より速いインスタンスは情報を待つように強制されるとともに、自身が継続しうるように次に必要な情報が取得可能であるか否かを判断するのに自身のコンピューティング資源を費やすように強制される。これにより、再計算を伴う予測と引き続くバックアップに資源を費やすのを避けることが可能になる。

【0056】以上、本発明に従いモデル処理の様々な側面を説明した。分割されたモデルを分散処理する方法、および異なるコンピューティング・ユニットで処理されるサブモデルを結合かつ制御する手法を提示した。ここで説明した様々な手法は単一システム、同種システム、および異機種システムに適用することができる。一例として、同じ物理マシン上の異なるハードウェア・シミュレータ群によって複数のサブモデルを処理することができる。

【0057】本発明はたとえばコンピュータ利用可能な媒体を備えた製品（たとえば少なくとも1つのコンピュータ・プログラム製品）中に含めることができる。上記媒体はその中にたとえば本発明の機能を提供するとともに容易にするコンピュータ読み取り可能なプログラム・コード手段を組み込んでいる。上記製品はコンピュータ・システムの一部に含めることもできるし、別売することもできる。

【0058】さらに、本発明によれば、マシンによって読み取り可能であるとともに、マシンによって実行可能であり本発明の機能を実行する命令群から成る少なくとも1つのプログラムを有形的に記録した少なくとも1つのプログラム記憶装置を提供することができる。

【0059】ここに示したフローチャートは単なる例にすぎない。本発明の本旨のうちで、これらのフローチャートまたはここで説明したステップ（すなわち操作）群には多くの変形例がありうる。たとえば、ステップ群は異なる順序で実行しうるし、あるいはステップを付加、削除、または変形することができる。これらの変形例はすべて特許請求の範囲の一部をなすものと考えられる。

【0060】以上、ここでは好適な実施形態を示すとともに説明したけれども、当業者にとって明らかなように、本発明の本旨のうちで様々な変形、付加、置換などをなすことができる。したがって、これらのものは特許請求の範囲で定義された本発明の範囲の内にあると考えられる。

【0061】まとめとして以下の事項を開示する。

(1) 分割されたモデルの、第1のサブモデルおよび第

2のサブモデルを含む複数のサブモデルを処理する方法であって、分割されたモデルの第1のサブモデルと第2のサブモデルとを結合するステップであって、前記第1のサブモデルがインタフェースする第1のカプラーを準備するステップと、前記第2のサブモデルがインタフェースする第2のカプラーを準備するステップと、前記第1のカプラーと前記第2のカプラーとをインタフェースさせるステップであって、前記第1のカプラーと前記第2のカプラーとの間にバッファを準備するステップとを備えたステップと、前記第1のサブモデルおよび前記第2のサブモデルを一括して処理するステップであって、前記第1のサブモデルおよび前記第2のサブモデルが前記第1のカプラー、前記バッファ、および前記第2のカプラーを用いて通信する、ステップとを備えたステップを備えた方法。

(2) 前記モデルが論理設計である、上記(1)に記載の方法。

(3) 前記一括して処理するステップが前記第1のサブモデルおよび前記第2のサブモデルを分散処理するステップを備え、前記分散処理するステップが分散プロセス制御を備えている、上記(1)に記載の方法。

(4) 前記一括して処理するステップがさらに前記第1のサブモデルを第1のシミュレータ・インスタンス上で処理するステップと前記第2のサブモデルを第2のシミュレータ・インスタンス上で処理するステップとを備え、前記第1のシミュレータ・インスタンスは前記第2のシミュレータ・インスタンスと異なり、前記第1のシミュレータ・インスタンスは前記バッファを用いて前記第2のシミュレータ・インスタンスと通信する、上記

(1)に記載の方法。

(5) 前記第1のシミュレータ・インスタンスは第1のコンピューティング・ユニットに常駐し、前記第2のシミュレータ・インスタンスは第2のコンピューティング・ユニットに常駐する、上記(4)に記載の方法。

(6) 少なくとも1つのサブモデルおよび付随するカプラーが少なくとも1つのマップされたデータ・ポートおよびクロック/サイクル・ポートを備え、前記クロック/サイクル・ポートは少なくとも1つのクロック/サイクル信号を供給し、前記少なくとも1つのクロック/サイクル信号は前記少なくとも1つのサブモデルと付随するカプラーとの間で前記少なくとも1つのマップされたデータ・ポートを通じてデータを転送するのに使用される、上記(1)に記載の方法。

(7) 前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方がトップ・レベル・カプラー・エンティティとカプラー共有ライブラリとを備え、前記第1のカプラーおよび前記第2のカプラーのうちの前記少なくとも一方が前記トップ・レベル・カプラー・エンティティを用いて前記第1のサブモデルおよび前記第2のサブモデルと個別に接続し、前記トップ・レベル・カプ

ラー・エンティティが少なくとも1つの入力と、少なくとも1つの出力と、前記第1のサブモデルおよび前記第2のサブモデルのうちの少なくとも一方の前記クロック/サイクル・ポートにマップされた複数のポートとを備え、前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方が前記カプラー共有ライブラリを用いて前記カプラーと前記バッファとの間でデータを転送する、上記(6)に記載の方法。

(8) 前記バッファが通信媒体を備え、前記通信媒体は前記第1のカプラーと前記第2のカプラーとの間でデータを渡すとともに監視する少なくとも1つのプロトコルを備えている、上記(1)に記載の方法。

(9) 前記一括して処理するステップがさらに、前記第1のカプラーが前記第1のサブモデルから取得したデータを前記バッファに送信し、前記第2のカプラーが前記データを前記バッファから取得し、前記データを前記第2のサブモデルに供給するステップと、前記送信と前記取得とを対応させることにより、前記データの一貫性を保証するステップとを備えた、上記(1)に記載の方法。

(10) 前記一貫性を保証するステップが、前記データにチェックサム値を挿入するステップと、前記第1のカプラーおよび前記第2のカプラーのうちの一方が前記バッファ中の前記データを調べて前記データの正確性を判断するステップと、前記データが不正確な場合、前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方が、前記データの前記バッファへの再送信および前記データの前記バッファからの再読み取りのうちの少なくとも一方を行なうステップとを備えた、上記

(9)に記載の方法。

(11) 前記挿入するステップが、前記チェックサム値を計算するステップを備え、前記計算するステップが、少なくとも1つの論理操作を前記データに適用するステップを備えている、上記(10)に記載の方法。

(12) 前記調べるステップが、前記データの新たなチェックサム値を計算するステップと、前記新たなチェックサム値と前記チェックサム値とを比較するステップとを備えた、上記(10)に記載の方法。

(13) 前記一貫性を保証するステップがさらに、前記バッファに、前記第1のカプラーが書き込みアクセス可能であるとともに前記第2のカプラーが読み取りアクセス可能な第1の一連ファイルを準備するステップと、前記バッファに、前記第1のカプラーが読み取りアクセス可能であるとともに前記第2のカプラーが書き込みアクセス可能な第2の一連ファイルを準備するステップと、所定の順序で、前記第1のカプラーが前記第1の一連ファイルに書き込み、前記第2のカプラーが前記第1の一連ファイルから読み取るステップと、前記所定の順序で、前記第2のカプラーが前記第2の一連ファイルに書き込み、前記第1のカプラーが前記第2の一連ファイル

から読み取るステップとを備えた、上記（９）に記載の方法。

（１４）前記インタフェースさせるステップがさらに、前記バッファ中のデータを監視して監視統計を供給するステップを備え、前記監視統計が前記第１のサブモデルおよび前記第２のサブモデルを統計的に再分割して前記一括処理を最適化するのを容易にする、上記（１）に記載の方法。

（１５）前記モデルがVHDLで表されている、上記（１）に記載の方法。

（１６）分割されたモデルの、第１のサブモデルおよび第２のサブモデルを含む複数のサブモデルを処理するシステムであって、分割されたモデルの第１のサブモデルと第２のサブモデルとを結合する手段であって、前記第１のサブモデルがインタフェースする第１のカプラーを準備する手段と、前記第２のサブモデルがインタフェースする第２のカプラーを準備する手段と、前記第１のカプラーと前記第２のカプラーとをインタフェースさせる手段であって、前記第１のカプラーと前記第２のカプラーとの間にバッファを準備する手段とを備えた手段と、前記第１のサブモデルおよび前記第２のサブモデルを一括して処理する手段であって、前記第１のサブモデルおよび前記第２のサブモデルが前記第１のカプラー、前記バッファ、および前記第２のカプラーを用いて通信する、手段とを備えた手段を備えた方法。

（１７）前記モデルが論理設計である、上記（１６）に記載のシステム。

（１８）前記一括して処理する手段が前記第１のサブモデルおよび前記第２のサブモデルを分散処理する手段を備え、前記分散処理する手段が分散プロセス制御を備えている、上記（１６）に記載のシステム。

（１９）前記一括して処理する手段がさらに前記第１のサブモデルを第１のシミュレータ・インスタンス上で処理する手段と前記第２のサブモデルを第２のシミュレータ・インスタンス上で処理する手段とを備え、前記第１のシミュレータ・インスタンスは前記第２のシミュレータ・インスタンスと異なり、前記第１のシミュレータ・インスタンスは前記バッファを用いて前記第２のシミュレータ・インスタンスと通信する、上記（１６）に記載のシステム。

（２０）前記第１のシミュレータ・インスタンスは第１のコンピューティング・ユニットに常駐し、前記第２のシミュレータ・インスタンスは第２のコンピューティング・ユニットに常駐する、上記（１９）に記載のシステム。

（２１）少なくとも１つのサブモデルおよび付随するカプラーが少なくとも１つのマップされたデータ・ポートおよびクロック／サイクル・ポートを備え、前記クロック／サイクル・ポートは少なくとも１つのクロック／サイクル信号を供給し、前記少なくとも１つのクロック／

サイクル信号は前記少なくとも１つのサブモデルと付随するカプラーとの間で前記少なくとも１つのマップされたデータ・ポートを通じてデータを転送するのに使用される、上記（１６）に記載のシステム。

（２２）前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方がトップ・レベル・カプラー・エンティティとカプラー共有ライブラリとを備え、前記第１のカプラーおよび前記第２のカプラーのうちの前記少なくとも一方が前記トップ・レベル・カプラー・エンティティを用いて前記第１のサブモデルおよび前記第２のサブモデルと個別に接続し、前記トップ・レベル・カプラー・エンティティが少なくとも１つの入力と、少なくとも１つの出力と、前記第１のサブモデルおよび前記第２のサブモデルのうちの少なくとも一方の前記クロック／サイクル・ポートにマップされた複数のポートとを備え、前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が前記カプラー共有ライブラリを用いて前記カプラーと前記バッファとの間でデータを転送する、上記（２１）に記載のシステム。

（２３）前記バッファが通信媒体を備え、前記通信媒体は前記第１のカプラーと前記第２のカプラーとの間でデータを渡すとともに監視する少なくとも１つのプロトコルを備えている、上記（１６）に記載のシステム。

（２４）前記一括して処理する手段がさらに、前記第１のカプラーが前記第１のサブモデルから取得したデータを前記バッファに送信し、前記第２のカプラーが前記データを前記バッファから取得し、前記データを前記第２のサブモデルに供給する手段と、前記送信と前記取得とを対応させることにより、前記データの一貫性を保証する手段とを備えた、上記（１６）に記載のシステム。

（２５）前記一貫性を保証する手段が、前記データにチェックサム値を挿入する手段と、前記第１のカプラーおよび前記第２のカプラーのうちの一方が前記バッファ中の前記データを調べて前記データの正確性を判断する手段と、前記データが不正確な場合、前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が、前記データの前記バッファへの再送信および前記データの前記バッファからの再読み取りのうちの少なくとも一方を行なう手段とを備えた、上記（２４）に記載の方法。

（２６）前記挿入する手段が、前記チェックサム値を計算する手段を備え、前記計算する手段が、少なくとも１つの論理操作を前記データに適用する手段を備えている、上記（２５）に記載のシステム。

（２７）前記調べる手段が、前記データの新たなチェックサム値を計算する手段と、前記新たなチェックサム値と前記チェックサム値とを比較する手段とを備えた、上記（２５）に記載のシステム。

（２８）前記一貫性を保証する手段がさらに、前記バッファに、前記第１のカプラーが書き込みアクセス可能で

10

20

30

40

50

あるとともに前記第2のカプラーが読み取りアクセス可能な第1の一連ファイルを準備する手段と、前記バッファに、前記第1のカプラーが読み取りアクセス可能であるとともに前記第2のカプラーが書き込みアクセス可能な第2の一連ファイルを準備する手段と、所定の順序で、前記第1のカプラーが前記第1の一連ファイルに書き込み、前記第2のカプラーが前記第1の一連ファイルから読み取る手段と、前記所定の順序で、前記第2のカプラーが前記第2の一連ファイルに書き込み、前記第1のカプラーが前記第2の一連ファイルから読み取る手段とを備えた、上記(24)に記載のシステム。

(29) 前記インタフェースさせる手段がさらに、前記バッファ中のデータを監視して監視統計を供給する手段を備え、前記監視統計が前記第1のサブモデルおよび前記第2のサブモデルを統計的に再分割して前記一括処理を最適化するのを容易にする、上記(16)に記載のシステム。

(30) 前記モデルがVHDLで表されている、上記(16)に記載のシステム。

(31) 分割されたモデルの、第1のサブモデルおよび第2のサブモデルを含む複数のサブモデルを処理するシミュレーション・システムであって、分割されたモデルの第1のサブモデルと第2のサブモデルとを、前記第1のサブモデルにインタフェースした第1のカプラーおよび前記第2のサブモデルにインタフェースした第2のカプラーを準備し、前記第1のカプラーと前記第2のカプラーとの間に設けたバッファによって前記第1のカプラーと前記第2のカプラーとをインタフェースさせることにより、結合するように適合した少なくとも1つのコンピューティング・ユニットを備え、前記少なくとも1つのコンピューティング・ユニットがさらに前記第1のサブモデルおよび前記第2のサブモデルを一括して処理するように適合しており、前記第1のサブモデルおよび前記第2のサブモデルが前記第1のカプラー、前記第2のカプラー、および前記第2のカプラーを用いて通信するシミュレーション・システム。

(32) 前記少なくとも1つのコンピューティング・ユニットが複数のシミュレータ・インスタンスを備え、前記複数のシミュレータ・インスタンスが第1のシミュレータ・インスタンスおよび第2のシミュレータ・インスタンスを含み、前記第1のシミュレータ・インスタンスが前記第1のサブモデルを処理し、前記第2のシミュレータ・インスタンスが前記第2のサブモデルを処理し、前記第1のシミュレータ・インスタンスが前記第2のシミュレータ・インスタンスとは異なる、上記(31)に記載のシミュレーション・システム。

(33) 前記第1のシミュレータ・インスタンスが第1のコンピューティング・ユニットに常駐し、前記第2のシミュレータ・インスタンスが第2のコンピューティング・ユニットに常駐する、上記(32)に記載のシミュ

レーション・システム。

(34) 分割されたモデルの、第1のサブモデルおよび第2のサブモデルを含む複数のサブモデルを処理する方法であって、分割されたモデルの第1のサブモデルと第2のサブモデルとを結合するステップであって、前記第1のサブモデルがインタフェースする第1のカプラーを準備するステップと、前記第2のサブモデルがインタフェースする第2のカプラーを準備するステップと、前記第1のカプラーと前記第2のカプラーとをインタフェースさせるステップであって、前記第1のカプラーと前記第2のカプラーとの間にバッファを準備するステップとを備えたステップと、前記第1のサブモデルおよび前記第2のサブモデルを一括して処理するステップであって、前記第1のサブモデルおよび前記第2のサブモデルが前記第1のカプラー、前記バッファ、および前記第2のカプラーを用いて通信する、ステップとを備えたステップを備えた方法を実行する、機械によって実行可能な命令から成る少なくとも1つのプログラムを有形的に記録した、機械読み取り可能なプログラム記憶装置。

(35) 前記モデルが論理設計である、上記(34)に記載のプログラム記憶装置。

(36) 前記一括して処理するステップが前記第1のサブモデルおよび前記第2のサブモデルを分散処理するステップを備え、前記分散処理するステップが分散プロセス制御を備えている、上記(34)に記載のプログラム記憶装置。

(37) 前記一括して処理するステップがさらに前記第1のサブモデルを第1のシミュレータ・インスタンス上で処理するステップと前記第2のサブモデルを第2のシミュレータ・インスタンス上で処理するステップとを備え、前記第1のシミュレータ・インスタンスは前記第2のシミュレータ・インスタンスと異なり、前記第1のシミュレータ・インスタンスは前記バッファを用いて前記第2のシミュレータ・インスタンスと通信する、上記

(34)に記載のプログラム記憶装置。

(38) 前記第1のシミュレータ・インスタンスは第1のコンピューティング・ユニットに常駐し、前記第2のシミュレータ・インスタンスは第2のコンピューティング・ユニットに常駐する、上記(37)に記載のプログラム記憶装置。

(39) 少なくとも1つのサブモデルおよび付随するカプラーが少なくとも1つのマップされたデータ・ポートおよびクロック/サイクル・ポートを備え、前記クロック/サイクル・ポートは少なくとも1つのクロック/サイクル信号を供給し、前記少なくとも1つのクロック/サイクル信号は前記少なくとも1つのサブモデルと付随するカプラーとの間で前記少なくとも1つのマップされたデータ・ポートを通じてデータを転送するのに使用される、上記(34)に記載のプログラム記憶装置。

(40) 前記第1のカプラーおよび前記第2のカプラー



のうちの少なくとも一方がトップ・レベル・カプラー・エンティティとカプラー共有ライブラリとを備え、前記第1のカプラーおよび前記第2のカプラーのうちの前記少なくとも一方が前記トップ・レベル・カプラー・エンティティを用いて前記第1のサブモデルおよび前記第2のサブモデルと個別に接続し、前記トップ・レベル・カプラー・エンティティが少なくとも1つの入力と、少なくとも1つの出力と、前記第1のサブモデルおよび前記第2のサブモデルのうちの少なくとも一方の前記クロック/サイクル・ポートにマップされた複数のポートとを備え、前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方が前記カプラー共有ライブラリを用いて前記カプラーと前記バッファとの間でデータを転送する、上記(39)に記載のプログラム記憶装置。

(41) 前記バッファが通信媒体を備え、前記通信媒体は前記第1のカプラーと前記第2のカプラーとの間でデータを渡すとともに監視する少なくとも1つのプロトコルを備えている、上記(34)に記載のプログラム記憶装置。

(42) 前記一括して処理するステップがさらに、前記第1のカプラーが前記第1のサブモデルから取得したデータを前記バッファに送信し、前記第2のカプラーが前記データを前記バッファから取得し、前記データを前記第2のサブモデルに供給するステップと、前記送信と前記取得とを対応させることにより、前記データの一貫性を保証するステップとを備えた、上記(34)に記載のプログラム記憶装置。

(43) 前記一貫性を保証するステップが、前記データにチェックサム値を挿入するステップと、前記第1のカプラーおよび前記第2のカプラーのうちの一方が前記バッファ中の前記データを調べて前記データの正確性を判断するステップと、前記データが不正確な場合、前記第1のカプラーおよび前記第2のカプラーのうちの少なくとも一方が、前記データの前記バッファへの再送信および前記データの前記バッファからの再読み取りのうちの少なくとも一方を行なうステップとを備えた、上記(42)に記載のプログラム記憶装置。

(44) 前記挿入するステップが、前記チェックサム値を計算するステップを備え、前記計算するステップが、少なくとも1つの論理操作を前記データに適用するステップを備えている、上記(43)に記載のプログラム記憶装置。

(45) 前記調べるステップが、前記データの新たなチェックサム値を計算するステップと、前記新たなチェックサム値と前記チェックサム値とを比較するステップとを備えた、上記(43)に記載のプログラム記憶装置。

(46) 前記一貫性を保証するステップがさらに、前記バッファに、前記第1のカプラーが書き込みアクセス可能であるとともに前記第2のカプラーが読み取りアクセス可能な第1の一連ファイルを準備するステップと、前

記バッファに、前記第1のカプラーが読み取りアクセス可能であるとともに前記第2のカプラーが書き込みアクセス可能な第2の一連ファイルを準備するステップと、所定の順序で、前記第1のカプラーが前記第1の一連ファイルに書き込み、前記第2のカプラーが前記第1の一連ファイルから読み取るステップと、前記所定の順序で、前記第2のカプラーが前記第2の一連ファイルに書き込み、前記第1のカプラーが前記第2の一連ファイルから読み取るステップとを備えた、上記(42)に記載のプログラム記憶装置。

(47) 前記インタフェースさせるステップがさらに、前記バッファ中のデータを監視して監視統計を供給するステップを備え、前記監視統計が前記第1のサブモデルおよび前記第2のサブモデルを統計的に再分割して前記一括処理を最適化するのを容易にする、上記(34)に記載のプログラム記憶装置。

(48) 前記モデルがVHDLで表されている、上記(34)に記載のプログラム記憶装置。

#### 【図面の簡単な説明】

【図1】 本発明の諸側面に従いサブモデル群を一括して処理する通信環境の一実施形態を示す図である。

【図2】 (a) 本発明の一側面に従ってシミュレートする1つのモデルの一実施形態を示す図である。(b) 図2(a)のモデルの分割を示す図である。(c) 本発明の一側面に従い結果のサブモデルの一例を示す図である。

【図3】 本発明の一側面に従い汎用カプラーおよびそのサブモデルへのインタフェースの一例を示す図である。

【図4】 本発明の一側面に従いVHDLを用いて表したサブモデルの一例を示す図である。

【図5】 本発明の一側面に従いVHDLを用いて表した汎用カプラー・エンティティの一例を示す図である。

【図6】 本発明の一側面に従いVHDLトップ・レベル・エンティティの一例を示す図である。

【図7】 本発明の一側面に従いカプラー共有ライブラリが出力をファイルに書き込むのに使用する論理の一例を示す図である。

【図8】 本発明の一側面に従いカプラー共有ライブラリが再送信機能を実行するのに使用する論理の一例を示す図である。

【図9】 本発明の一側面に従いカプラー共有ライブラリがファイルから入力を読み取るのに使用する論理の一例を示す図である。

【図10】 本発明の一側面に従いカプラー共有ライブラリがデータの再送信を要求するのに使用する論理の一例を示す図である。

#### 【符号の説明】

100 コンピューティング・ユニット

102 コンピューティング・ユニット



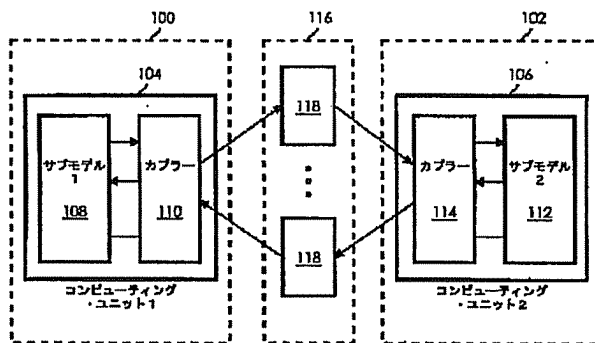
104 ライセンスを受けたハードウェア・シミュレータのインスタンス  
 106 ライセンスを受けたハードウェア・シミュレータのインスタンス  
 108 サブモデル  
 110 カプラー  
 112 サブモデル  
 114 カプラー  
 116 バッファ  
 118 複数ファイル  
 200 未分割のモデル  
 202 動作コンポーネント  
 204 クロック/サイクル・エンティティ  
 206 サブモデル  
 208 サブモデル  
 210 サブモデル1の動作コンポーネント  
 212 サブモデル2の動作コンポーネント  
 214 クロック/サイクル・エンティティ  
 216 チャネル

\* 218 入出力  
 222 入力  
 224 入力  
 220 出力  
 226 出力  
 228 クロック/サイクル・ポート  
 230 クロック/サイクル・ポート  
 300 サブモデル  
 302 カプラー  
 10 304 トップ・レベル・カプラー・エンティティ  
 306 VHDLへのAPI  
 308 カプラー共有ライブラリ  
 310 出力  
 312 入力  
 314 クロック/サイクル・ポート  
 316 出力ポート  
 318 入力ポート  
 320 クロック/サイクル・ポート

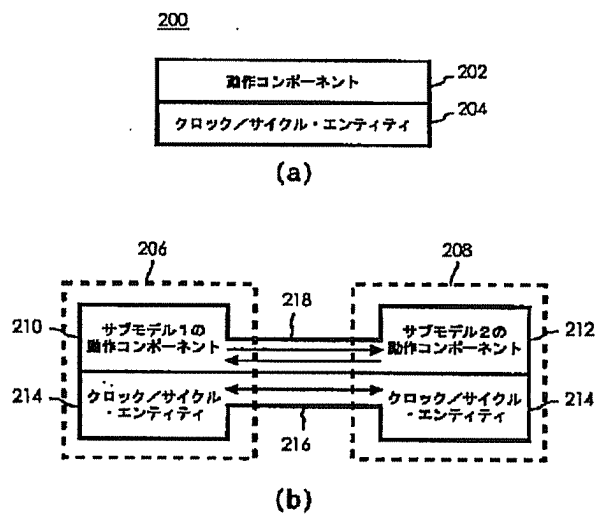
\*

20

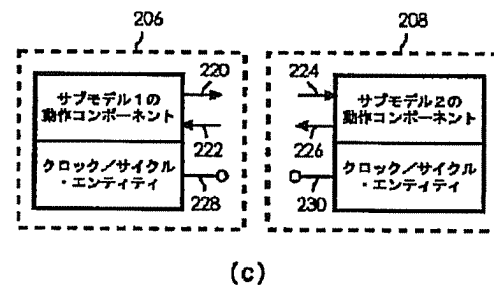
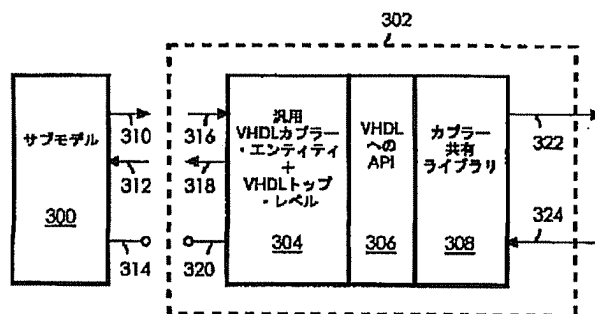
【図1】



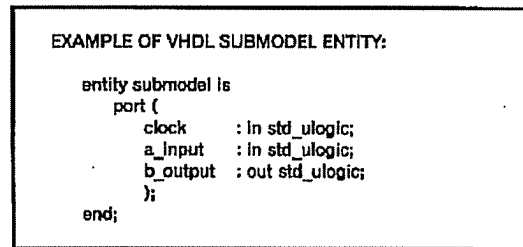
【図2】



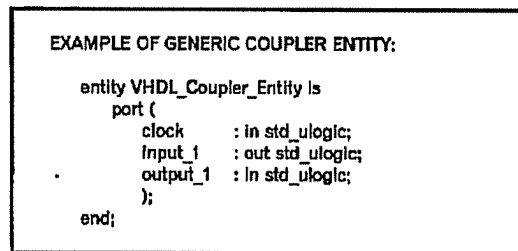
【図3】



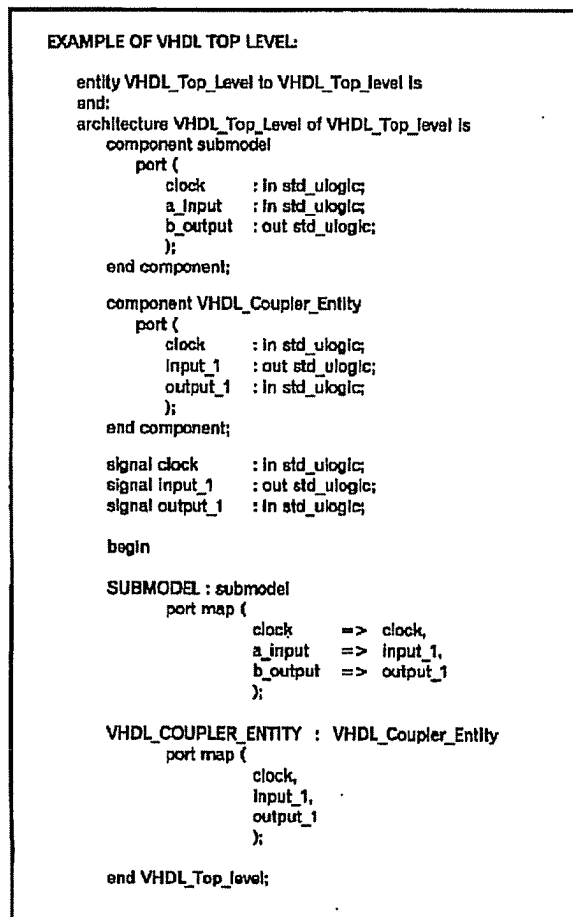
【図 4】



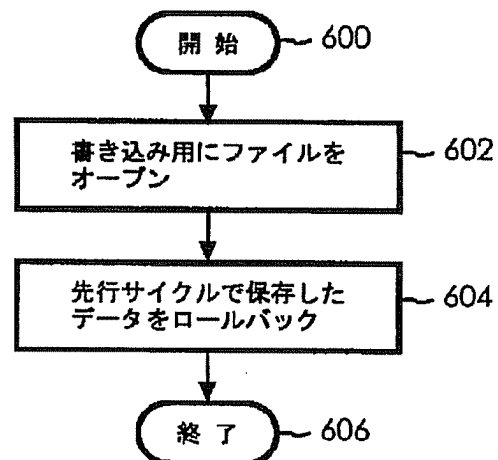
【図 5】



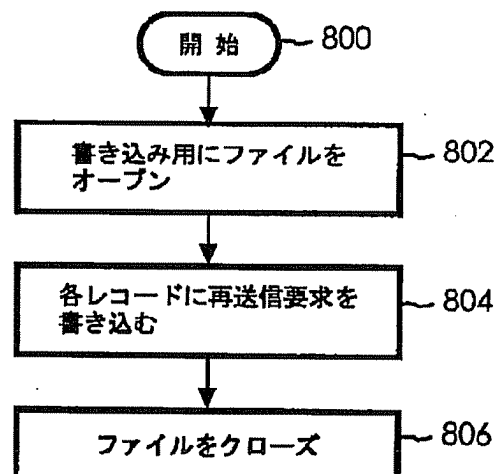
【図 6】



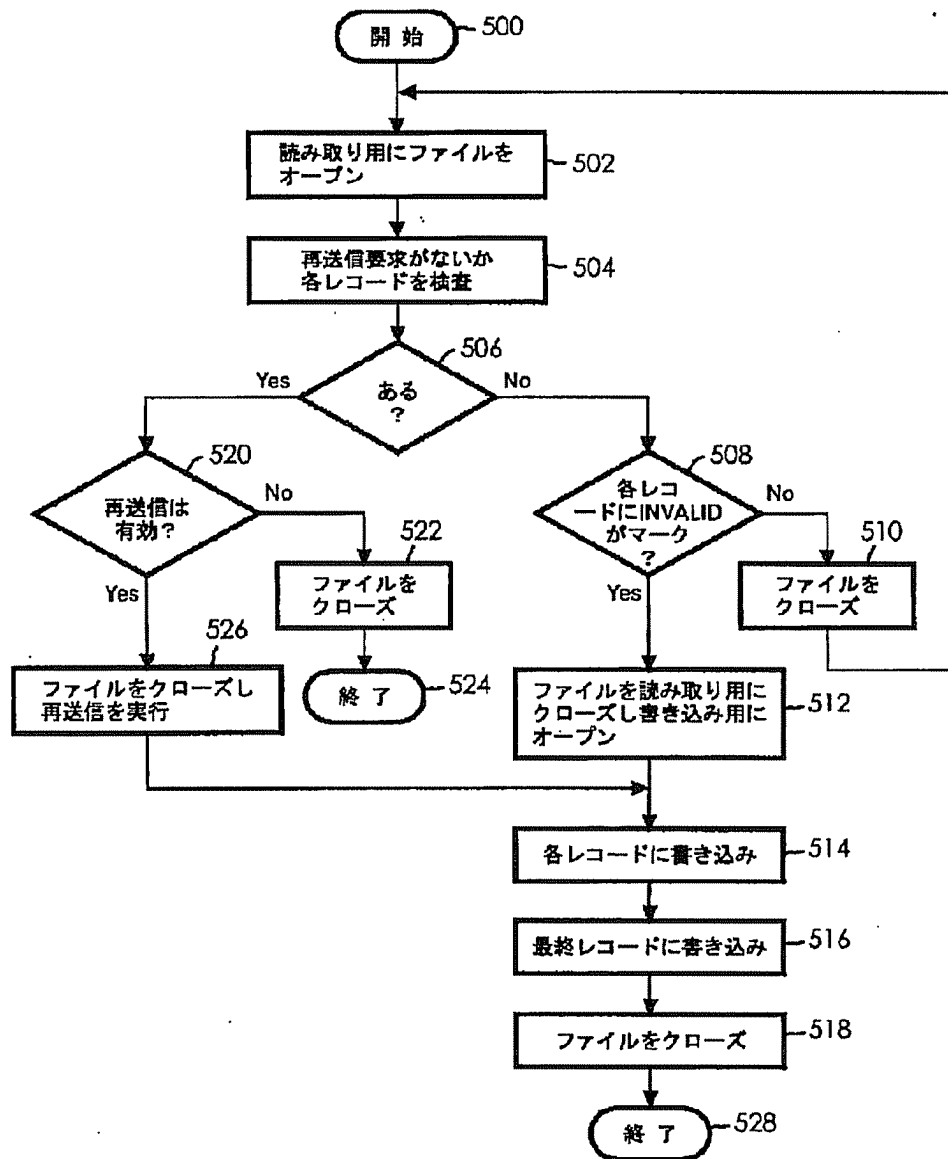
【図 8】



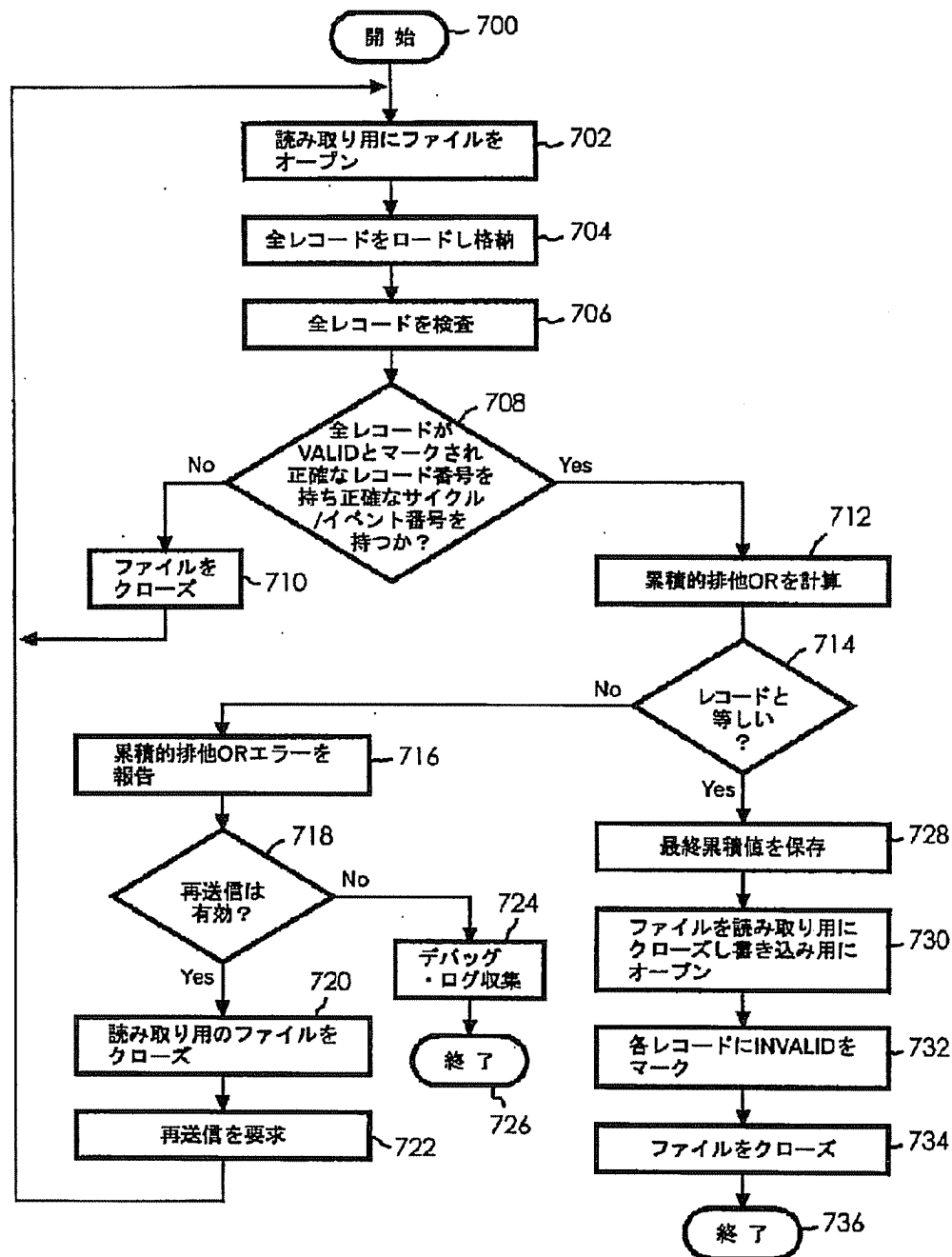
【図 10】



【図 7】



【図9】



フロントページの続き

(72)発明者 ウィリアム・ケイ・メローズ  
アメリカ合衆国 ニューヨーク州 12514,  
クリントン コーナーズ、トレッスル レ  
ーン 67

(72)発明者 マーヴィン・ジェイ・リッチ  
アメリカ合衆国 ニューヨーク州 12603,  
ポキプシー、バード レーン 7  
Fターム(参考) 5B046 AA08 BA03 CA06 JA05